Challenging Aerospace Problems for Intelligent Systems

K. KrishnaKumar[1], J. Kanashige[1], A. Satyadas[2]
[1]NeuroEngineering Laboratory
NASA Ames Research Center
kkumar, jkaneshige@mail.arc.nasa.gov

[2]IBM Corporation, USA.
antony_satyadas@us.ibm.com

1 Abstract

In this paper we highlight four problem domains that are well suited and challenging for intelligent system technologies. The problems are defined and an outline of a probable approach is presented. No attempt is made to define the problems as test cases. In other words, no data or set of equations that a user can code and get results are provided. The main idea behind this paper is to motivate intelligent system researchers to examine problems that will elevate intelligent system technologies and applications to a higher level.

2 Introduction

Intelligent System (IS) applications have gained popularity among aerospace professionals in the last decade due to the ease with which several of the IS tools can be implemented. The applications have gained popularity among both the technical and user communities for both intellectual curiosity and for practical reasons. Some of the novel ideas using IS include spacecraft autonomy, aircraft control, modeling, airfoil design, satellite operations, missile design, vehicle health management, and so on. In the next several sections, we present some problem domains that are challenging to achieve using intelligent system technologies. If successful, the ensuing IS approaches can revolutionize many aspects of aerospace applications. In each of the problem domains discussed, we first present the potential problem area and outline one IS configuration that could help achieve success. The problem domains covered include:

- Automated design
- Intelligent maneuvering
- Smart agent society
- Real-time optimization

3 Automated Design

3.1 The Problem

Aerospace systems of tomorrow will be complex and their design interdisciplinary. For example, design optimization conducted individually on subsystems such as wing, propulsion, and automatic control will not integrate without extensive and costly redesign. A unified design that integrates all facets of a system is difficult if not impossible to find using current optimization techniques. Traditional design approaches rely on cut-and-try approaches adopted by designers that can be conducted very well using high performance computers. The bottleneck for computer implementation is the

lack of (1) a universal representation of the design features and (2) a procedure for the design features to be cut-and-tried *by a computer* in an optimal way. There are other potential problems associated with computer-based automated design. These are:

- Time required for a system simulation is large prohibiting use of full fidelity simulation of the problem.
- Analytical derivatives of the objectives of design are frequently unavailable and numerical gradients are expensive to compute.
- Design space consists of both continuous and/or discrete parameters
- Design response surface is non-linear, discontinuous, or undefined in some regions. Several local extrema is common in many applications.
- Initial guesses for the design are costly and might lead to inaccessible solution spaces.

The desire here is to use intelligent system technologies in an unified way to arrive a design framework by which automated design can be achieved using computers.

3.2 A Solution

The idea of letting the computer do the cut-and-try is not new to the world of optimization. A genetic optimization search basically involves a cut-and-try approach that is driven by the survival-of-the-fittest concept. If universal representations (standardized representation) can be developed, then a technique like genetic algorithms could be used. The other challenge is the speed at which designs could be evaluated. This implies that modeling techniques such as neural networks, fuzzy systems and so on can play an important role here.

To obtain a universal representation, we introduce the concept of a design building block. A design building block is a way to represent a feature as an input-output function with tunable parameters. These functions can be polynomials, pieces of a neural network, look-up tables, etc. These building blocks are identified first and a library of these building blocks is constructed. This library is continuously updated as more and more designs are created. This library is then used to construct design solutions for existing problems using variants of a genetic algorithm or other combinatorial optimization problem.

We see several critical benefits of this approach to the design community. These are:

1. Library of Design building blocks: Library techniques are commonly used in modern integrated circuit (IC) design packages. This greatly reduces the number of combinations with which the designer must deal, thereby speeding up the search process. The design building blocks concept tailors this idea to the specific design.
2. Innovation is achieved by the use of evolutionary search with stochastic operators. This greatly reduces the chance of reaching a local solution and solves many of the problems associated with gradient search techniques.
3. Long-term memory can be provided to the design package to remember good design solutions for later use via the use of the micro-features of the immune system, a variant of a genetic search process [1-3].

In summary, one needs the following to achieve automated design:

- an universal representation scheme for representing the building blocks of the design
- an appropriate genetic coding to accommodate the building block concept
- a class of performance criteria and utility functions

Figure 1 presents the overall architecture of the system proposed.

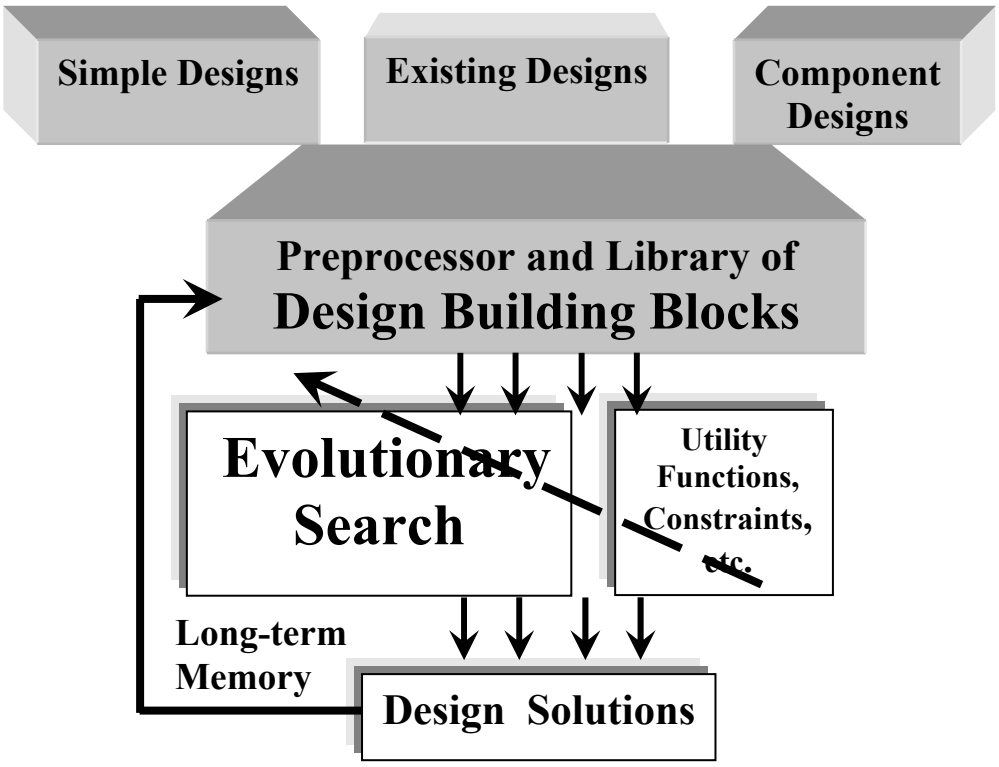


Figure 1. A Concept Diagram for Immunized Design Optimization.

Simple Design Solutions: Many engineering problems have simplified models and solutions that are routinely used as a first guess approximation of the complete solution. These solutions could be a rule-of-thumb solution or mathematically optimized using simpler models. These solutions are then converted to design building blocks to be used in the evolutionary search. Examples include:

- Linearized models and linear solutions for non-linear control problems.
- Knowledge base of operator rules, etc.

Existing Design Solutions: Many designs are in a way an improvement over existing designs. Although this is not true across the whole spectrum, at least the new designs use subsets of the old system components. These existing designs can then be converted to design building blocks and used in the search. Examples include:

- Available airfoil shapes for aerodynamic shape optimization
- Available nozzle shapes for Nozzle design.

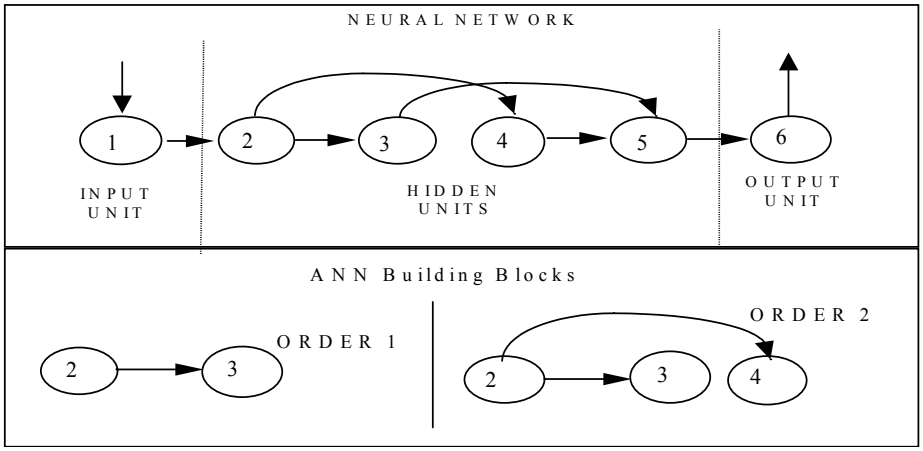
Component Design Solutions: When sub-components interact, the design solutions are easier if they are considered as non-interacting. These design solutions will not be the optimal solutions when interactions are considered but provide a means of arriving at certain fundamental building blocks. Examples include:

- In simultaneous actuator-sensor placement and control system optimization problem, one can design the placement separate from control system design.
- Another example is in aerodynamic-propulsion interaction in all-attitude control problems with thrust vectoring. Once again, the aerodynamic control system can be designed independent of the propulsion control system and converted to design building blocks.

Library of Design Building Blocks: Design building blocks are defined as segments of a design solution that contribute in establishing a good fit to the requirements of the design. Design Building blocks can be of different order. The order of a building block specifies the number of specific modules (or parts or some pre-defined configurations) in a building block. Determining important building blocks is problem dependent and both *a priori* knowledge and the ability to identify through genetic search will be useful here. Once the building blocks are identified, certain preprocessing and statistical analysis can be carried out to enable a faster search.

Next, we present two examples of design building blocks we have defined in our earlier studies [1-3].

- **Neural Network Building Blocks:** Examples of building blocks using neural connections are shown below. The building blocks are specified using a representation scheme that uses a neuron as the basic processing element. Thus the *Order 1* building block consists of two neurons and the relationship between them. In the case of Neural Networks, the relationship is the connection strength and the neurons are characterized by their type (input, hidden, output), the type of aggregation, and activation functions.



- **Aerodynamic shape optimization Building Blocks:** For airfoil optimization, satisfactory results have been obtained using an array of Bezier curves for the definition of the parameterized shapes. A Bezier polynomial of order *n* is defined by

$$b(t) = \sum_{i=0}^{n} P_i \Phi_i(t)$$

where

$$\Phi_i(t) = \left(\frac{n!}{i!(n-1)!} \right) t^i (1-t)^{n-1}$$

P_i's ar the vertices of the Bezier control polygon. Given the above parameterization, the design building blocks can be represented by P_i or a family of P_i's. Similar to neural network building blocks, *Order 1* building block will be just one P_i, *Order 2* will be two P_i's and so on.

Utility Functions, etc. To evaluate the designs produced by the genetic search, a series of performance measures and constraints have to be defined. Due to the ability of evolutionary algorithms to search through non-continuous spaces, both traditional and nontraditional measures could be easily combined.

4 Intelligent Maneuvering

Over the past several years, various adaptive control techniques have been developed which are capable of accommodating a wide range of damage or failure conditions [4-6]. These approaches apply techniques, such as neural networks, fuzzy logic, and parameter identification, to improve aircraft stability and control under varying conditions. While these approaches address the continuous-time aspects of "how to control" an aircraft, they do not address the discrete-time strategic and tactical decision-making aspects of "how to fly" an aircraft. The outer-loop control and flight planning portions of flight are normally left to conventional autopilots and waypoint-following flight management systems.

4.1 The Problem

Current levels of automation allow pilots to assign direct tasks to automatic systems, such as autopilots and flight management systems. These automated systems have been used in commercial aircraft for a number of years. While their design can incorporate many aspects of a pilot's experience, they do not possess the reasoning or learning abilities of a pilot. As a result, pilots are still responsible for supervising the performance of these systems as well as providing direction in the event of required changes. By applying intelligent methods of automation, pilots, ground-based operators, or autonomous executives can defer the responsibilities from performing and supervising tasks, to focus on managing goals and objectives.

4.2 A Solution

In order to make reliable decisions when flying aircraft under varying conditions, intelligent technologies must be applied which are capable of responding to changing goals and objectives, while taking correction actions in the presence of internal and external events. Multiple methods, such as heuristics, artificial intelligence concepts, and other soft computing techniques, can be applied in order to replace the experience, reasoning and learning abilities of pilots. Just as pilots use different mental approaches

when performing various tasks, different intelligent automation techniques can be applied which correspond to the computational nature and time restrictions associated with those tasks. One method of organizing various intelligent techniques is to establish a tiered architecture, with separate deliberative and reactive decision-making layers.

System Architecture: In terms of achieving a flight-path goal, a pilot's behavior can be captured through a layered model consisting of discrete-time strategic planning and tactical maneuvering, and continuous-time manual control (Figure 2) [7]. The discrete nature of strategic and tactical behaviors allows for automated decision-making techniques to be applied. Furthermore since strategic planning decisions are less time-critical, more computationally intensive approaches can be utilized. All of the real-time processing elements can be isolated in the automation of manual control.

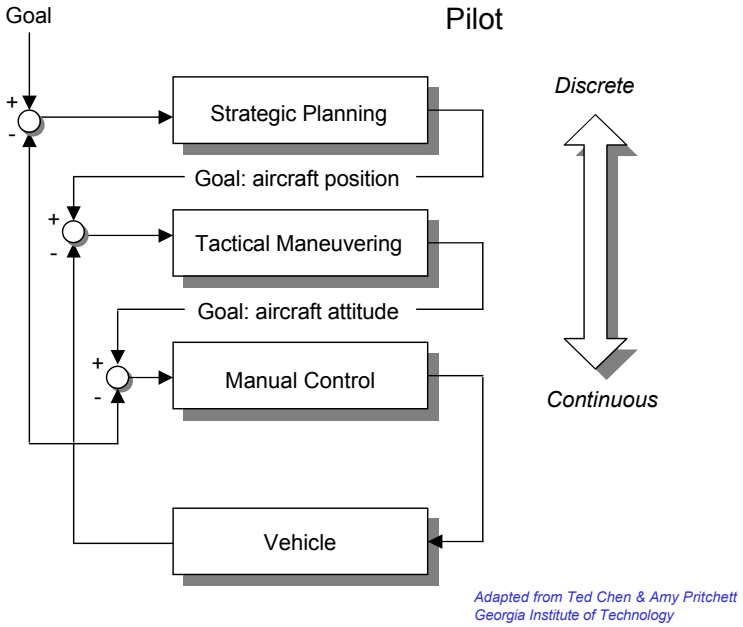


Figure 2. Pilot Behavior Hierarchy

Figure 3 shows the resulting conceptual integrated architecture. Strategic technologies would perform longer-term flight planning, in order to meet dynamic mission goals and objectives, while avoiding obstacles and staying within performance boundaries. Tactical technologies would perform time-critical flight path operations, including aggressive maneuvers in the presence of unexpected obstacles. Conventional and adaptive control techniques would be used to automate the manual control task of the pilot, through the automated selection of flight modes and targets.

Strategic Planning: From a pilot's point of view, any flight can be thought of as a plan of turns, descents, and other discrete actions. These actions alter the continuous flight-path or aircraft trajectory, until desired goals are reached. Actions are not limited to merely changes in the aircraft speed and orientation. Some actions also change the aircraft configuration itself, such as extension of flaps and gears or the dumping of excess fuel. Various trajectory specialists could be used to produce candidate flight path segments,

representing the "experience" of a pilot. The selection criteria would be based on mission goals and constraints provided by a pilot, ground-based operator, or autonomous executive.

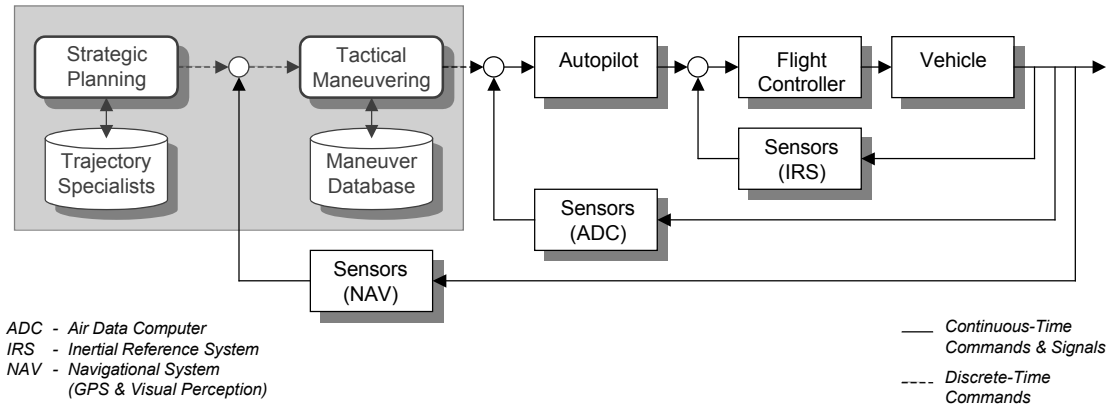


Figure 3. A Concept Diagram for an Integrated Architecture

Tactical Maneuvering: At the highest level, the pilot compares the commanded flight-path with that of the current aircraft and selects the maneuvers capable of achieving that command. Pilots use their knowledge of aircraft capabilities and of near-optimal maneuvering strategies in order to select the necessary actions. Various methods could be used to select the necessary maneuvers. A maneuver database, also representing the "experience" of a pilot, could be used to provide pre-canned, or automatically generated, maneuvering elements and established sequences. Vehicle models can be used to provide the necessary predictive information for decision-making, representing the equivalent of a pilot's "understanding" of the internal performance of the aircraft. Appropriate flight modes and targets would be sent to the autopilot system, when necessary to initiate the desired actions.

Automatic Controls: Conventional control techniques can be used to automate the continuous-time control task of the pilot. However, various adaptive control techniques can also be used to provide the "learning" ability of a pilot. These techniques have the potential of improving handing qualities, and thereby increasing the accuracy of simplified closed-loop models used for decision-making.

5 Society of Prediction Agents

Prediction or forecasting is concerned with using the knowledge of present and past events to make calculated estimates of future events. Prediction is an universal phenomenon used in low level cognitive tasks such as vision, and perception, and high level cognitive tasks such as planning and making inferences. The inherent subjectivity, randomness, domain dependencies, and uncertainties makes the prediction problem extremely difficult and challenging to the scientist and the engineer. Moreover, often there are multiple sources of prediction with varying degrees of reliability and confidence [8,10,11,12,14]. There are two distinct prediction techniques, namely qualitative and quantitative. Qualitative techniques are referred to as judgmental, technological, or non-

statistical and usually depend on expert opinion. Examples of such techniques include decision matrices, S-curves, game theory, systems analysis, Delphi method (jury of executive opinion method), and more recent fuzzy decision support expert systems [8]. Quantitative prediction techniques are based on the assumption of historical continuity. Options include time series, regression, and combinations of the two.

5.1 The Problem

In aeronautics applications prediction techniques vary from a simple need to predict an aerodynamic derivative to more complex situation such as fault predictions. In space applications, prediction applications are more complex. Since the information sources can be very varied, it is not sufficient to provide just human-like characteristics atop the prediction technologies, we need to a go a step further and provide the capability of a group of experts to arrive at a prediction. This task is quite challenging.

5.2 A Solution

More recent work related to a group prediction includes the Fuzzy Multiple Criteria Group Decision Making (FMCGDM) based prediction proposed by Satyadas [7] that introduces the notion of a group of expert agents selecting the appropriate prediction from a pool of predictions. This is a hybrid model with a collaboration framework at the highest level and various multi-sources of prediction at lower levels.

The FMCGDM equations can be defined as follows: Given **n** fuzzy rules

$R= [r_1, r_2,....., r_n]$, where

r_n= IF x_{n1} AND/OR x_{n2} AND/OR x_{nj} THEN y_{n1} AND $y_{n2}...y_{nk}$., each consisting of

j inputs/states/antecedents/premises $X=[x_1, x_2,\ldots x_j]$,
k outputs/actions/consequents/conclusions $Y=[y_1,y_2, ... y_k]$,

an universe of discourse resolution of *p* steps; the state vector

$S = [s_1, s_2, ...s_n]$, where

$s_n = x_{n1}$ AND/OR x_{n2} AND/OR...x_{nj} and the fuzzy action curve set

$A=[a_1,a_2,....a_n]$, where

$a_n = [z_{n1}, z_{n2}, ...z_{np}]$

is obtained by applying fuzzy implication on S over the universe of discourse. Appropriate fuzzy aggregation (example: Max) and defuzzification (example: Center of Area) algorithms can be applied on A to obtain a crisp output. The fuzzy membership function parameters and the rule structure can be learned using evolutionary algorithms. A weight matrix *W[j+k,i]* may be used to capture the weightage provided by *i* members of the group on the *j+k* criteria, a weight matrix *X[n,i]* may be used to identify the

importance of each rule. Techniques such as logarithmic regression or Saaty's AHP may be employed to apply the weights on to the fuzzy rules.

Given *u* sources of prediction, the above-described fuzzy system will provide a measure of importance for each of the prediction as its output. Appropriate ranking and selection techniques have to be used to compute the final prediction. The following technology issues need to be addressed:

1. Given that there are achievable multi dimensional levels of Smart Prediction Agents, what are the best set of building blocks that will enable easy implementation of these Predictors.
2. Once these building blocks are defined and developed, develop a set of intelligent prediction algorithms that integrate these building blocks. These shall be uniquely indexed using the Smart Prediction Agent (SPA) rating given in Table 1. The relevance and value of the algorithms are context dependent.
3. The implementation challenge will be to ensure component-based architecture and standards that will allow interoperability and meets non-functional requirements such as scalability, security, reliability, and the like.

The multi dimensional levels of smart prediction presented in [8] (See Table 1 below) provides a means for qualifying and quantifying smart prediction techniques. We believe that a practical way to define the predictive capabilities of a system is to approach it as having multi dimensional levels of capabilities for self-improvement, problem solving, knowledge and domain bounds, and trade-offs. The capabilities are additive towards higher levels.

Any smart prediction agent may be identified as SPA[*p,i,k,s,r*] where *p*=problem solving, *i*=self-improvement, *k*=knowledge building, *s*=severity/domain bounds, and *r*=risk management/trade-off. This enables a SPA rating for various prediction systems based on the values of *p, i, k, s*, and *r*.

Table 1. Levels of Dimensions for Smart Prediction

Level	Problem Solving	Self-Improvement	Knowledge Building	Severity	Risk Management/ Trade Offs
0	Uni-variate	Statistical	A priori	Crisp compute	Accuracy/Trend
1	Multi-Variate	Generalized	Derived	Imprecise	Local/distributed
2	Patterns/Cases	Adaptive	Inferred	Incomplete	Temporal
3	Stationarity	Optimized	Discovered	Subjective	Spatial
4	Multi-source	Plan		Complex	Static/Dynamic
5	Causality			Chaotic	Maximum/Avg

Problem Solving Dimension for Smart Prediction

Six levels have been identified in this dimension (Table 1). The uni-variate and multi-variate refers to the dependent variables. Patterns/cases with the associated behavior present a higher difficulty level for problem solving. Stationarity and ergodicity issues present the next level of challenge. A stationary process will have time-invariant mean and variance. The covariance between values of the process at two time points will depend only on the distance between these time points and not on time itself. The

ergodicity assumption requires that values of the process sufficiently far apart in time are almost un-correlated so that averaging a time series through time continually adds new and useful information to the average. This is followed by multiple sources of prediction that includes the system approach.

Self-Improvement Dimension for Smart Prediction

The levels identified have been motivated by the levels of intelligent control [9]. Level 0 relies on statistical information to improve the quality of prediction. The next level requires generalization capabilities that can be achieved using computational intelligence techniques such as artificial neural networks. This is followed by adaptive learning capabilities. Optimality of prediction, with the goal of minimization or maximization of a utility function over time, is addressed by Level 3. Coupling to a control module, that may require modified predictions, is introduced in the next level. Level 4 presents the planning aspects of self-improvement that provides the ability to perform predictions based on a plan with the associated goals and risk mitigation strategies.

Knowledge Building Dimension for Smart Prediction

Level 0 relies on a priori knowledge. Various techniques, from ad-hoc to well defined and complex, from artificial intelligence have been proposed by researchers. Techniques include memory modules (neural and otherwise) with fixed weights, fixed fuzzy rules and parameters, and suitable parameters for GA. The generic coding structure of GA is an attractive feature. Simple derivations based on observed facts are provided in Level 1. The next level has inference capabilities. Level 3 provide knowledge discovery capabilities that demand advanced data and text mining features. Learning plays a critical role here.

Severity Dimensions for Smart Prediction

The severity dimension provides a measure of the domain bounds. The levels start from crisp computation, imprecise and/or incomplete domains, and subjectivity. Level 4 represents a complex domain that is in the edge of chaos, and level 5 - a chaotic domain.

Risk Management Dimension for Smart Prediction

This dimension address trade-off that promotes risk management. Level 0 trade-off is between accurate prediction and just being able to predict the trend (whether it is upward or downward). The local/distributed aspect of the problem/solution space is introduced in Level 1. This is followed by temporal and spatial trade-off. Level 4 mitigates the risks between static and dynamic prediction. Choices between optimizing predicted values and just obtaining an average prediction is provided in Level 5.

We believe some of the challenges include:

- Challenges in introducing the upper layer that brings together prediction, control, and other capabilities.
- Challenges in cognition that will be required to be solved.
- Other aspects of the aircraft and linkages with the base - where adding smartness is valuable?

6 Real-time optimization under uncertainty

6.1 The Problem

Building large-scale intelligent solutions for optimal decision-making and control is of great importance to the advancement of operational autonomous systems. Werbos[15] has addressed an important formulation for achieving higher-order intelligence in these systems. The outlined approach extends the capabilities of adaptive critics[1] [16-18] to include temporal chunking. Although many researchers believe that real-time optimization over time and under uncertainty is a good approach to replicating higher order intelligence, the current applications of intelligent systems do not reflect the type of complexity processed by the mammalian brain.

6.2 The Solution

The central theme of the proposed solution is to arrive at a modular architecture for achieving higher order intelligent decision-making and control. To achieve this, we first define a MAC (Model-Action-Critic) neuron and extend it to represent a network of these with associated training signals. For effective autonomous decision making, we need adaptive control driven by an adaptive critic.

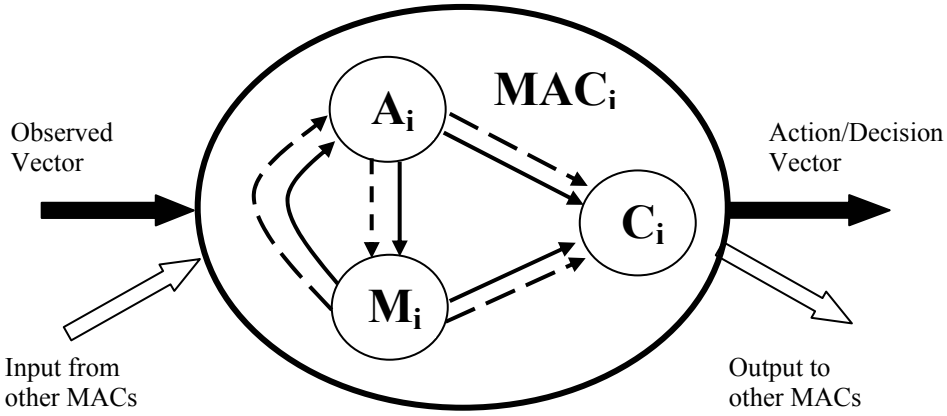


Figure 4 Model-Action-Critic module represented as one computational entity

[1] An adaptive critic adapts itself and at the same time it outputs a performance measure that can be used to update a controller network and/or a decision network. Howard's formulation of dynamic programming [19] is the inspiration behind the simplest version of adaptive critics, namely, Heuristic Dynamic Programming (HDP) critics.

$$J(x_t) = \underset{u_t}{Max}\left\{ U_{PM}(x_t) + \gamma J(x_{t+1}) \;\middle|\; x_{t+1} = f(x_t, u_t, noise) \right\}$$

where x_t is the state vector, u_t is the control vector, $U_{PM}(.)$ is the one stage Performance Measure function, $f(.,.,.)$ is the model of the system, and $\gamma\left(0 < \gamma \le 1\right)$ is the discount factor.

In Figure 4, we present a single MAC (Model-Action-Critic) neuron in which the interactions within three sub modules and with the external world (fat solid arrows) and other MACs (fat hollow arrows) are shown. These inputs are available to all the three modules (M, A, and C). The dashed-lines and the solid lines between the three sub modules denote information being passed for different time periods to keep in mind the temporal nature of the training.

Once we identify MAC as one computational entity, network of these critics can be built by specifying either the partitioning before hand (an example is given later) or defining the desired hierarchy (number of MAC neurons, number of layers, etc.). Once these decisions have been taken, the MAC neurons can be assembled as a network of critics.

Temporal Chunking for Brain-like Intelligence

Given a policy π (control or decision strategy) and a probability transition matrix $\underline{\underline{P}}$, one can write the Howard's formulation of the recursive equation as:

(2) $\underline{J}^{\pi} = \underline{U}^{\pi} + ((\underline{\underline{P}}^{\pi})^{T} / (1+r))(\underline{J}^{\pi})$

Now if substitute $\underline{\underline{M}}^{\pi} = (\underline{\underline{P}}^{\pi})^{T} / (1+r)$, we have

(3) $\underline{J}^{\pi} = \underline{U}^{\pi} + (\underline{\underline{M}}^{\pi} \underline{J}^{\pi})$

The above equation represents a fixed policy with just value updates. In a policy update, the control *u* (or decision *d*) is chosen to optimize (maximize or minimize) the right hand side of the equation.

In equation 3, if the matrix $\underline{\underline{M}}^{\pi}$ is sparse (which is usually the case), one can partition the state space into smaller blocks within which the transition probabilities are non-zero and transition from one block to the other happens only for exit and entry states. This partition could be defined *a priori* or learnt on-line (which is a much more difficult task). In some problems, as shown later, such partitions are well understood.

Now let us say we have two partitioned blocks *A* and *B*, with *A* representing the current block in which the system is operating and *B* representing a block to which the system could transition to. Starting from Block *A*, we have two transition probability matrices, P^A representing transition probability within block *A* and P^{AB} the probability of transition into block *B*. Similar to *M* defined earlier, one can define M^A and M^{AB} matrices and derive an equation similar to equation 3,

(5) $\underline{J}^{\pi}\big|_{A} = \underline{U}^{\pi}\big|_{A} + \underline{\underline{M}}^{A} \underline{J}^{\pi}\big|_{A} + \underline{\underline{M}}^{AB} (\underline{J}^{\pi}\big|_{B})$

In the above equation, the notation $\underline{V}\big|_{A}$ represents a portion of the variable vector $\underline{V}$ that applies to states within block *A*.

If we now extend the partition B to include more blocks, we have

(6) $$\underline{J}^{\pi}\Big|_A = \underline{U}^{\pi}\Big|_A + \underline{\underline{M}}^{A}\underline{J}^{\pi}\Big|_A + \sum_{B\in n(A)} \underline{\underline{M}}^{AB}(\underline{J}^{\pi}\Big|_B)$$

where $B \in n(A)$ represents the blocks in B that can be transitioned into by states in block A.

As shown by Werbos[15], the above equation can be written as,

(7) $$\underline{J}^{\pi}\Big|_A = \underline{J}^{A} + \sum_{B\in n(A)} \underline{\underline{J}}^{AB}(\underline{J}^{\pi}\Big|_B)$$

with $\underline{J}^{A} = (\underline{\underline{I}} - \underline{\underline{M}}^{A})^{-1}(\underline{U}^{\pi}\Big|_A)$ and $\underline{J}^{AB} = (\underline{\underline{I}} - \underline{\underline{M}}^{A})^{-1}\underline{\underline{M}}^{AB}$

Now one can write the following recursive relations similar to Bellman's

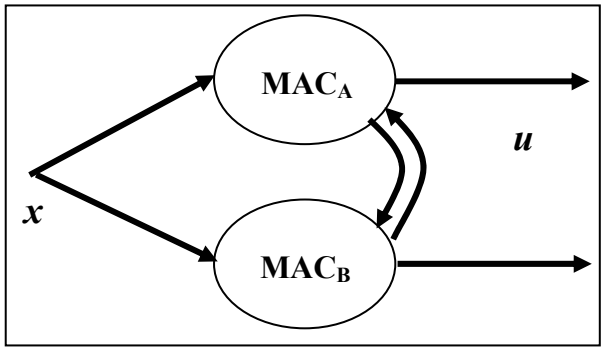
(8) $\underline{J}^{A} = \underline{U}^{\pi}\Big|_A + \underline{\underline{M}}^{A}\underline{J}^{A}$ and $\underline{J}^{AB} = \underline{\underline{M}}^{AB} + \underline{\underline{M}}^{A}\underline{J}^{AB}$

Using equations 8a and 8b in conjunction with a learning algorithm, $\underline{J}^{A}$ *and* $\underline{J}^{AB}$ can be updated and thus $\underline{J}$ can be calculated using equation 7. Finally, control (or decision) can be updated based on $\underline{J}$.

A simple example: We will define a simple example to illustrate the use of the idea presented. Let a system consist of two states and one control as shown below.

$\dot{x} = f_A(x_1, x_2, u)$ *for* $x_{21} \le x_2 \le x_{22}$
$\dot{x} = f_B(x_1, x_2, u)$ *for* $x_{22} < x_2 \le x_{23}$

where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $f_A = \begin{bmatrix} f_{A1} \\ f_{A2} \end{bmatrix}$, $f_B = \begin{bmatrix} f_{B1} \\ f_{B2} \end{bmatrix}$, and u is the control.



Note that the system has a clear partition based on the state x_2. Based on this pre-defined partition, one can design MAC$_A$ partition ($x_{21} \le x_2 \le x_{22}$) and MAC$_B$ partition ($x_{22} \le x_2 \le x_{23}$) and can be implemented as shown to the right.

Learning the underlying hierarchy: In many problems, the underlying partitions are known or can be conceptualized. For true brain-like intelligence, it will be desirable to learn such partitions. This could be achieved either off-line or on-line. Definitely on-line implementation will be very difficult to achieve. Off-line synthesis could be achieved in several ways. One simple way is to use parsimonious networks with pruning capabilities

to learn the internal partitions. Another approach could be clustering via unsupervised learning.

Decisions and failure accommodation

As suggested by Werbos and others, the partitions can be navigated more efficiently by including a decision layer in the network of critics. This concept is illustrated below. Also, failure accommodation could be handled inside of the decision layer. In the figure shown, we assume that the network behaves like a winner-all type with lateral inhibition implying that only one of the neurons output the control vector.

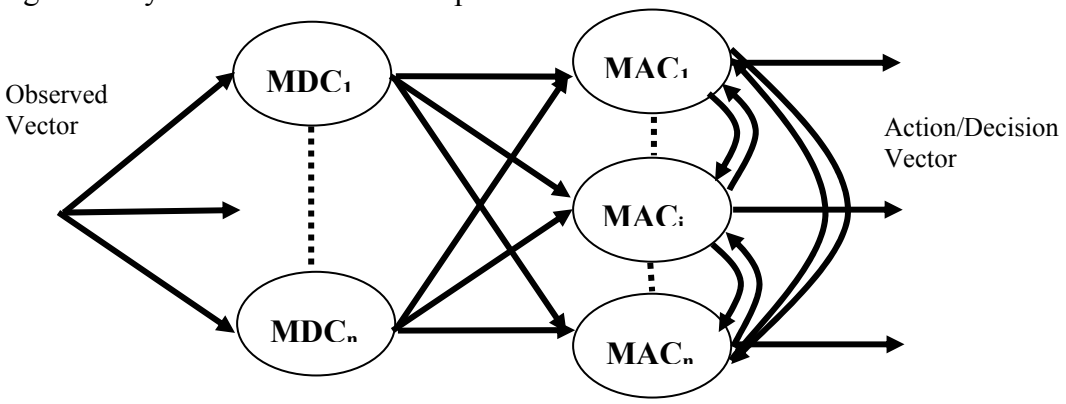


Figure 5. Network of Critics with a Decision Maker Layer
(MDC – Model-Decision-Critic)

7 Conclusions

In this paper, we presented four problem domains that are well suited for further advances in intelligent systems. These domains include, design, autonomous maneuvering, prediction, and decision-making. All of these require advances in intelligent system technologies to enable higher levels of automation, intelligence, and application success. All four problem domains defined are accompanied by one possible approach based on authors' knowledge and perspective. It is hoped that this will further advance intelligent system research and development for aerospace applications.

There are definitely several other problem domains where intelligent system technologies could help but need major advances to achieve high levels of success. Some of these areas include data-driven inverse design and modeling, autonomous planning and scheduling, and bio-inspired aerial vehicles.

8 References

1. K. KrishnaKumar, Immunized Neurocontrol: Concepts and Initial Results, Presented at the workshop on combinations of genetic algorithms and neural networks, COGANN'92, Baltimore, MD, June, 1992.
2. K. KrishnaKumar and J. C. Neidhoefer, Immunized Artificial Systems--Concepts and Applications, in Genetic Algorithms in Computers and Engineering, John Wiley & Sons, 1997.

3. K. KrishnaKumar and J. C. Neidhoefer, Immunized Neuro-control, Expert Systems with Applications, 1997.
4. J. Kaneshige and K. Gundy-Burlet, Integrated Neural Flight and Propulsion Control System, AIAA-2001-4386, August 2001.
5. R. T. Rysdyk and Anthony J. Calise, Fault Tolerant Flight Control via Adaptive Neural Network Augmentation, AIAA 98-4483, August 1998.
6. K. Krishnakumar, N. Kulkarni, "Inverse Adaptive Neuro-Control for the control of a turbofan engine", Proceedings of AIAA conference on Guidance, Navigation and Control, Portland, OR, 1999.
7. T. L. Chen, A. R. Pritchett, On-The-Fly Procedure Development for Flight Re-Planning Following System Failures, AIAA 2000-0300, January 2000.
8. A. Satyadas, Cognitive Prediction and Control using Soft Computing, Technical Report, University of Alabama, USA, 1998.
9. K. KrishnaKumar, Levels of Intelligent Control, AIAA Tutorial at New Orleans, LA, August 1997.
10. G. C. Reinsel, Element of multivariate time series analysis, Springer-Verlag, New York, 1986.
11. A. S. Weigend, N. A. Gershenfeld (editors), Time Series Prediction, Addison-Wesley Publishing Co., New York, 1994.
12. D. G. Bails, L. C. Peppers, Business fluctuations: forecasting techniques and applications, Prentice-hall, Inc., New Jersey, 1982.
13. U. Harigopal, A. Satyadas, Cognizant Enterprise Maturity Model, co-guest editors: A. Satyadas, U. Harigopal, N. Cassaigne, IEEE Transactions on Systems Man and Cybernetics - Part C: Applications and Reviews: Special Issue on Knowledge Management, Vol. 31, No. 4, pp 449-459, November 2001.
14. J. Hamilton, Time Series Analysis, Princeton University Press, Princeton, NJ, USA, 1994.
15. P. Werbos, A Brain-like Design to Learn Optimal Strategies in Complex Environments, in Brain-Like Computing and Intelligent Information Systems, Springer-Verlag Singapore Pte. Ltd. 1998
16. A. Barto, Reinforcement Learning and Adaptive Critic Methods. Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nosttrand Reinhold, Kentucky, 1992.
17. I. Bertsekas, J. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, Massachusetts, 1996.
18. P. Werbos, Approximate Dynamic Programming For Real-Time Control and Neural Modeling, Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nosttrand Reinhold, Kentucky, USA, 1993.
19. R. Howard. Dynamic Programming and Markov Process, Cambridge, MA, MIT Press, 1960.
20. K. Krishnakumar, Optimization of the Neural Net Connectivity Pattern Using a Back-Propagation Algorithm, Journal of Neurocomputing, (5) pp. 273-286, 1993.